

# Phun, geometrie perfette

(i tutorial di Alessandro de Simone)

Copyright Alessandro de Simone 2008 ([www.alessandrodesimone.net](http://www.alessandrodesimone.net)) - È vietato trascrivere, copiare, stampare, tradurre, riprodurre o divulgare il presente documento, anche parzialmente, senza l'autorizzazione scritta dell'autore. I siti Internet, le case editrici e le pubblicazioni di settore che intendano utilizzare questo documento possono contattare l'autore ([contatti@alessandrodesimone.net](mailto:contatti@alessandrodesimone.net)) per gli accordi del caso.

## Italiano (pag. 2)

Questo è il quarto tutorial dedicato a **Phun**: prima di intraprenderne la lettura si consiglia di consultare i tutorial precedenti:

- **Phun, primi passi** – Introduzione alle funzioni principali di Phun
- **Phun, giochiamo al biliardo** – Simulazione del gioco del biliardo
- **Posizionamento orizzontale perfetto** – Primi passi nella codifica dei file PHN

I Tutorial sono liberamente scaricabili dal mio sito (<http://www.alessandrodesimone.net/Phun/index.htm>) che vi consiglio di visitare frequentemente per verificare la presenza di eventuali aggiornamenti o nuovi Tutorial.

# Phun, perfect geometries

(i tutorial di Alessandro de Simone)

Copyright Alessandro de Simone 2008 ([www.alessandrodesimone.net](http://www.alessandrodesimone.net)) – No transcribing, no copyng, no reproducing, no translating, no printing, no publishing this document – even if partially – without author's written authorization. Websites and publishing house who wish to employ this document must write the author ([contatti@alessandrodesimone.net](mailto:contatti@alessandrodesimone.net)).

## English (pag. 5)

This is the fourth Tutorial about **Phun**: please, read the previous tutorials:

- **Phun, Getting started** – Intro to the basic Phun's functions (Italian language only).
- **Phun, Let's game billiards** – Game billiards simulation (Italian language only).
- **Perfect horizontal alignment** – Getting started on PHN files (Italian and English)

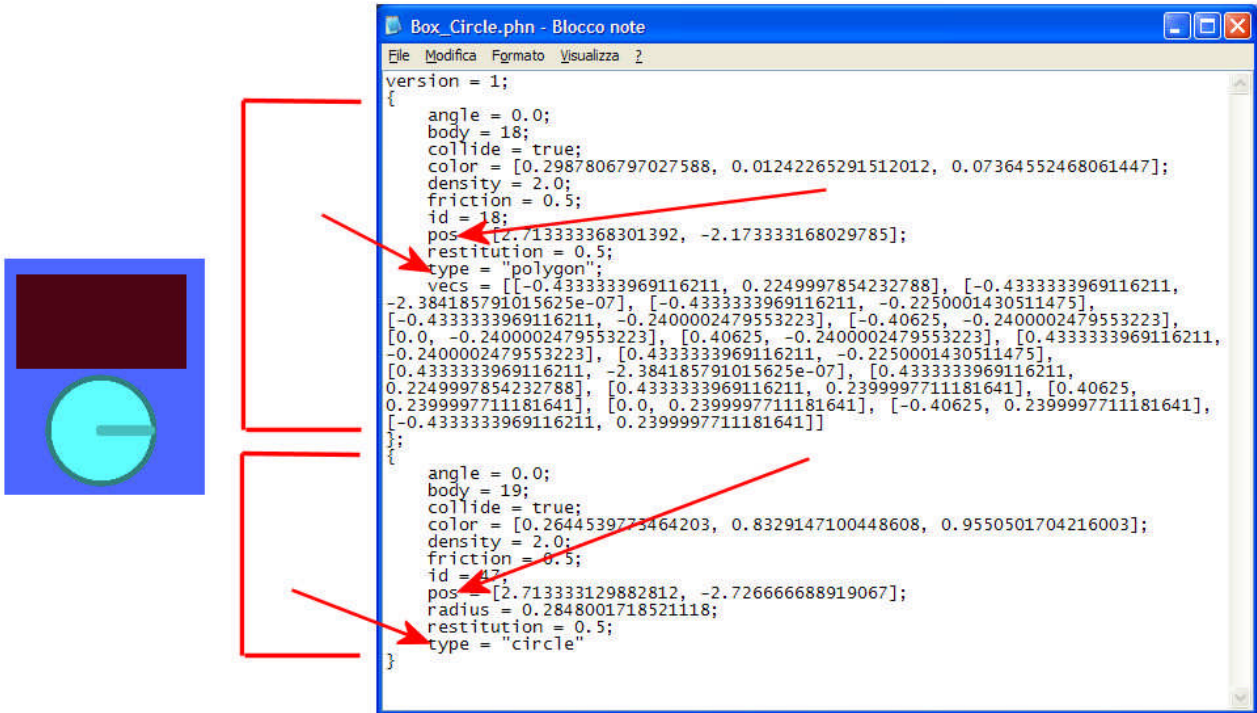
You can freely download the PDF's format tutorials from my website (<http://www.alessandrodesimone.net/Phun/index.htm>). Please, frequently visit my website to discover if there are new Tutorials or / and update.

## I codici di Phun

Gli oggetti che compaiono in una qualsiasi scena di **Phun** sono caratterizzati da un insieme di codici che, per nostra fortuna, sono in puro formato testo (ASCII).

Per seguire il presente *Tutorial*, disegnate un semplice rettangolo e un cerchio, più o meno sovrapposti, e registrate la scena assegnando il nome *Box\_Circle.PHN*.

Esaminando con **BloccoNote** il file creato noterete una situazione più o meno simile a quella della figura seguente.



E' facile individuare a occhio i due "blocchi" (indicati in figura dalle parentesi quadre in rosso), che del resto sono racchiusi ciascuno tra una coppia di parentesi graffe { } aperte e chiuse. A seconda della scena è possibile individuare altri codici. Nel caso della figura sono presenti i seguenti codici:

Codice	Valore	Commento
Angle	Valore decimale (pos./neg.)	Angolo di rotazione
Body	Positivo intero	???
Collide	True / False	Capacità di collisione
Color	Tre valori decimali positivi	Valori RGB (Red, Green, Blue)
Density	Valore decimale positivo	Peso
Friction	Valore decimale positivo	Attrito
id	Valore intero positivo	N. oggetto (???)
Pos	Coppia di valori decimali (pos./neg.)	Coordinate origine oggetto
Restitution	Valore decimale positivo	???
Type	Polygon / Circe / Spring ...	Tipologia oggetto
Vecs	Valori decimali (pos./neg.)	Coordinate vertici oggetto
Radius	Valore decimale positivo	Solo per cerchi

Di alcuni codici è intuitivo il significato, mentre l'interpretazione di altri (contrassegnati dai tre punti di domanda "???) è piuttosto difficoltosa. Rimane il fatto che è possibile intervenire direttamente sul file PHN (meglio se su una sua copia), modificando tali valori

ma stando bene attenti a non alterare o cancellare inavvertitamente parametri vitali: il programma potrebbe all'inizio funzionare correttamente, salvo poi andare in crash.

## Eccesso di parametri

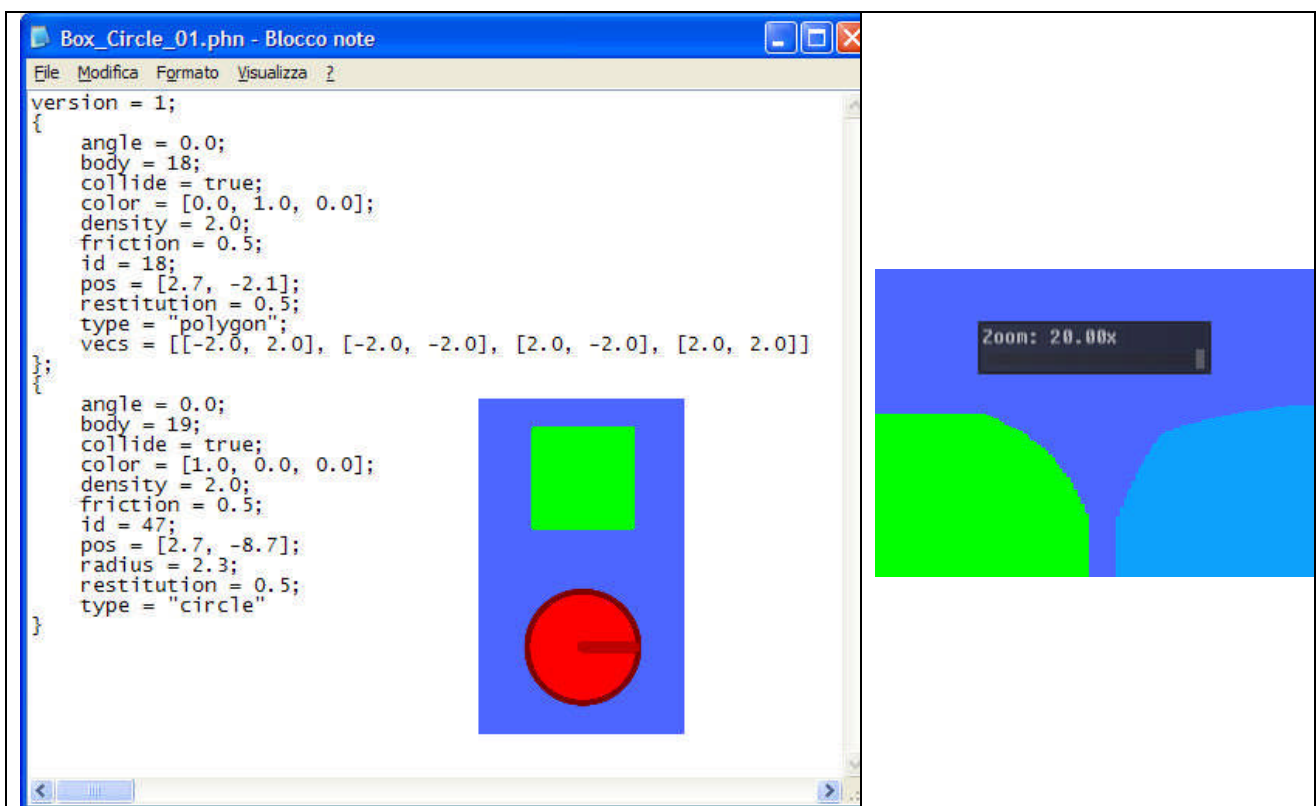
Esaminando il "listato" della figura precedente è facile rendersi conto che il cerchio è caratterizzato dai due classici valori che contraddistinguono una qualunque circonferenza: le coordinate dell'**origine** (coppia di valori **POS**) e **diametro** (**Radius**).

Il rettangolo, invece, *dovrebbe* essere contraddistinto da quattro coppie di valori, corrispondenti alle coordinate dei suoi quattro vertici. Il numero di parametri presenti è invece ben maggiore.

Effettuando vari esperimenti ho scoperto alcuni particolari:

- I valori numerici decimali, ove compaiono, sono in doppia precisione. Il numero di cifre presenti dopo la virgola deriva dalla necessità, per *Phun*, di individuare il posizionamento del puntatore del mouse quando l'utente disegna (esempio: vertici di una figura), assegna un colore (inevitabili imprecisioni dovute allo spostamento manuale del cursore), posizionamento dell'oggetto (trascinamento e spostamento).
- *Phun* accetta comunque valori decimali con soltanto una o due cifre dopo la virgola.
- *Phun* accetta rettangoli caratterizzati da quattro coordinate soltanto.

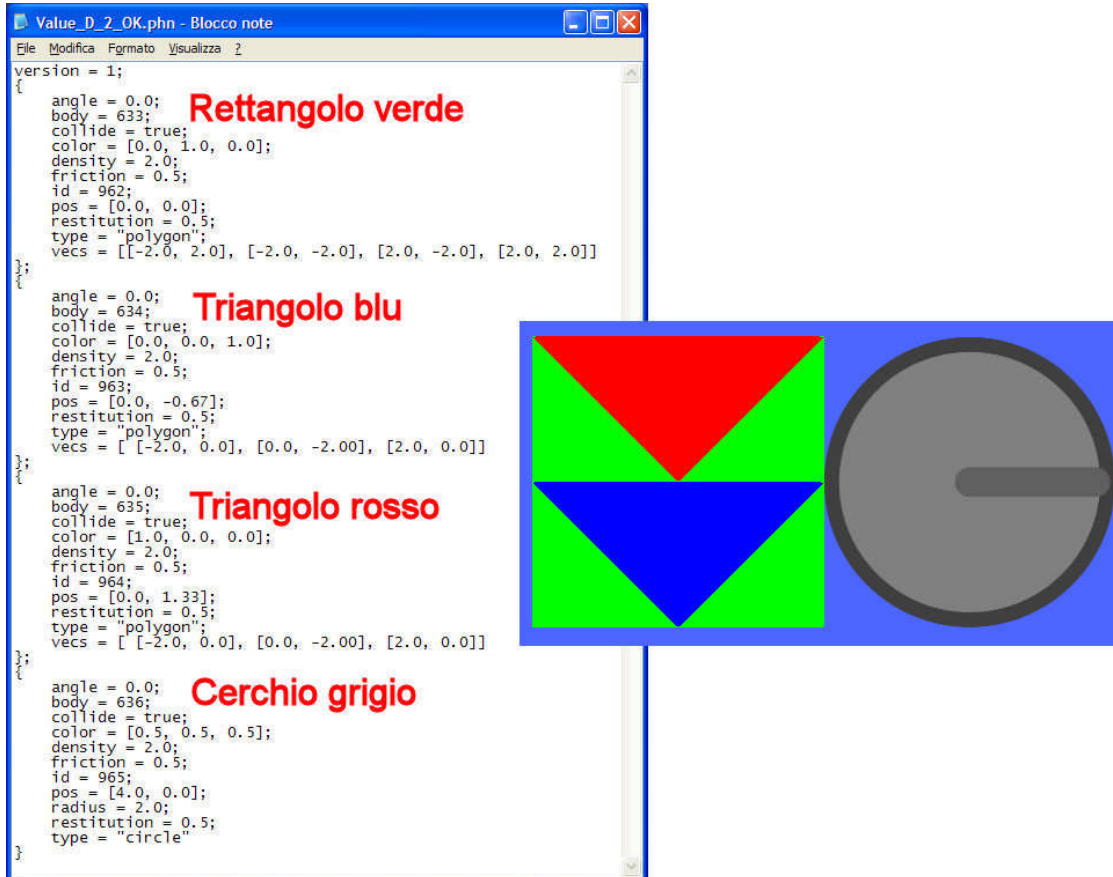
La parte sinistra della figura che segue, e che deriva direttamente modificando a mano il listato della figura precedente, dimostra che *Phun* accetta valori più "semplici" e, soprattutto, che è capace di gestire rettangoli caratterizzati solo da quattro coordinate.



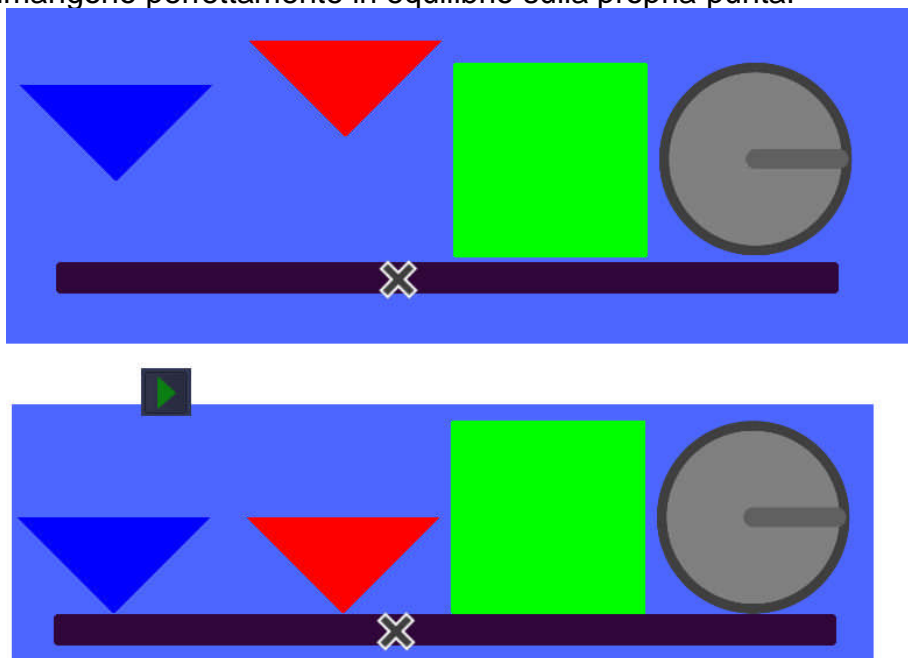
Che differenza, allora, c'è tra i rettangoli creati digitando direttamente i valori dei vertici nel file e quelli che invece disegniamo con lo strumento *Box* di *Phun*? Esaminando il notevole ingrandimento della figura a destra è possibile notare la differenza tra il vertice del rettangolo verde di sinistra (i cui parametri sono: **vecs = [[-2.0, 2.0], [-2.0, -2.0], [2.0, -2.0], [2.0, 2.0]]**) e quello celeste di destra, i cui parametri **vecs** sono invece "ridondanti".

## Nuovi orizzonti

La possibilità di digitare direttamente valori specifici nei file *PHN* offre inaspettate occasioni per sperimentare nuove forme. La figura che segue dimostra come sia possibile creare **triangoli** perfettamente simmetrici e posizionati con un rigore impossibile da ottenere altrimenti.



Ma c'è di più: la perfetta simmetria, impostabile digitando valori estremamente precisi, consente un effetto collaterale inaspettato: nella scena precedente, per esempio, i due triangoli sono talmente equilibrati che, spostandoli in orizzontale, sollevandoli e lasciandoli precipitare, rimangono perfettamente in equilibrio sulla propria punta!



# Phun, perfect geometries

(i tutorial di Alessandro de Simone)

Copyright Alessandro de Simone 2008 ([www.alessandrodesimone.net](http://www.alessandrodesimone.net)) – No transcribing, no copyng, no reproducing, no translating, no printing, no publishing this document – even if partially – without author’s written authorization. Websites and publishing house who wish to employ this document must write the author ([contatti@alessandrodesimone.net](mailto:contatti@alessandrodesimone.net)).

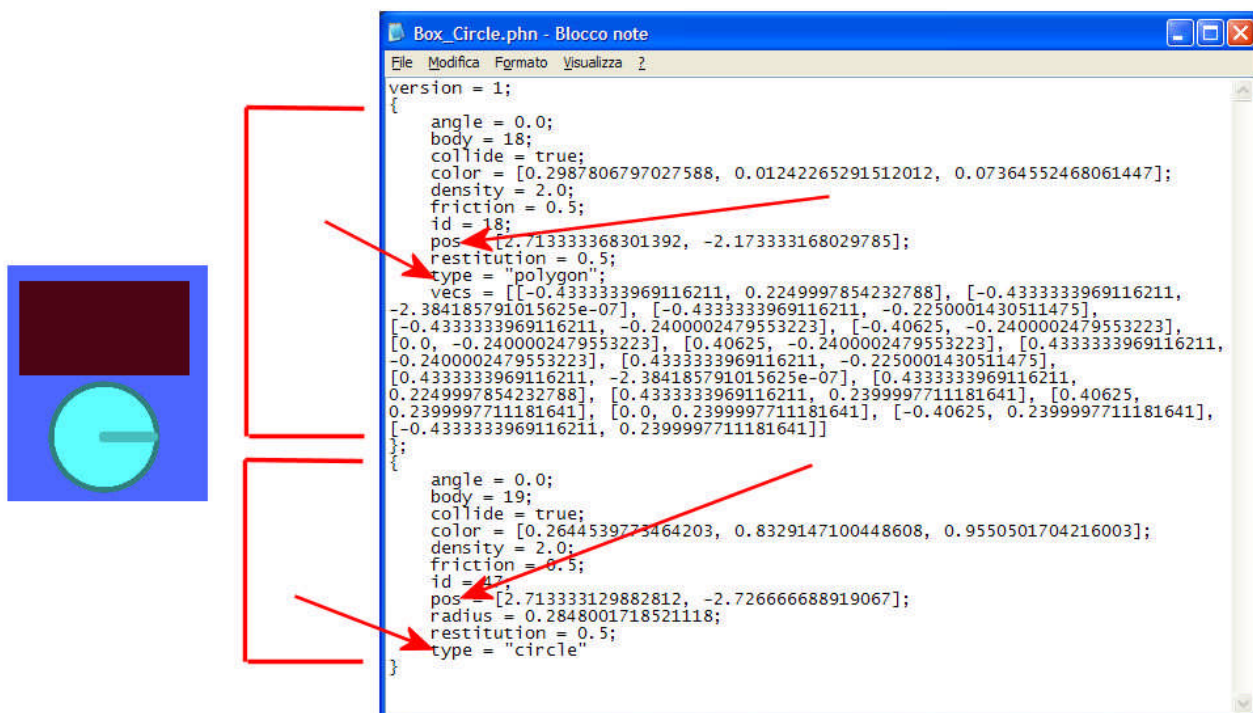
This is the fourth Tutorial about **Phun**: please, read the previous tutorials:

- **Phun, Getting started** – Intro to the basic Phun’s functions (Italian language only).
- **Phun, Let’s game billiards** – Game billiards simulation (Italian language only).
- **Perfect horizontal alignment** – Getting started on PHN files (Italian and English)

You can freely download the PDF’s format tutorials from my website (<http://www.alessandrodesimone.net/Phun/index.htm>). Please, frequently visit my website to discover if there are new Tutorials or / and update.

## Phun’s codes

Phun’s objects are a simple “group” of codes. Try to design a rectangle and a circle and save the scene with name *Box\_Circle.PHN*. If you examine the file with an ASCII text editor (like **NotePad**) you’ll se something like this:



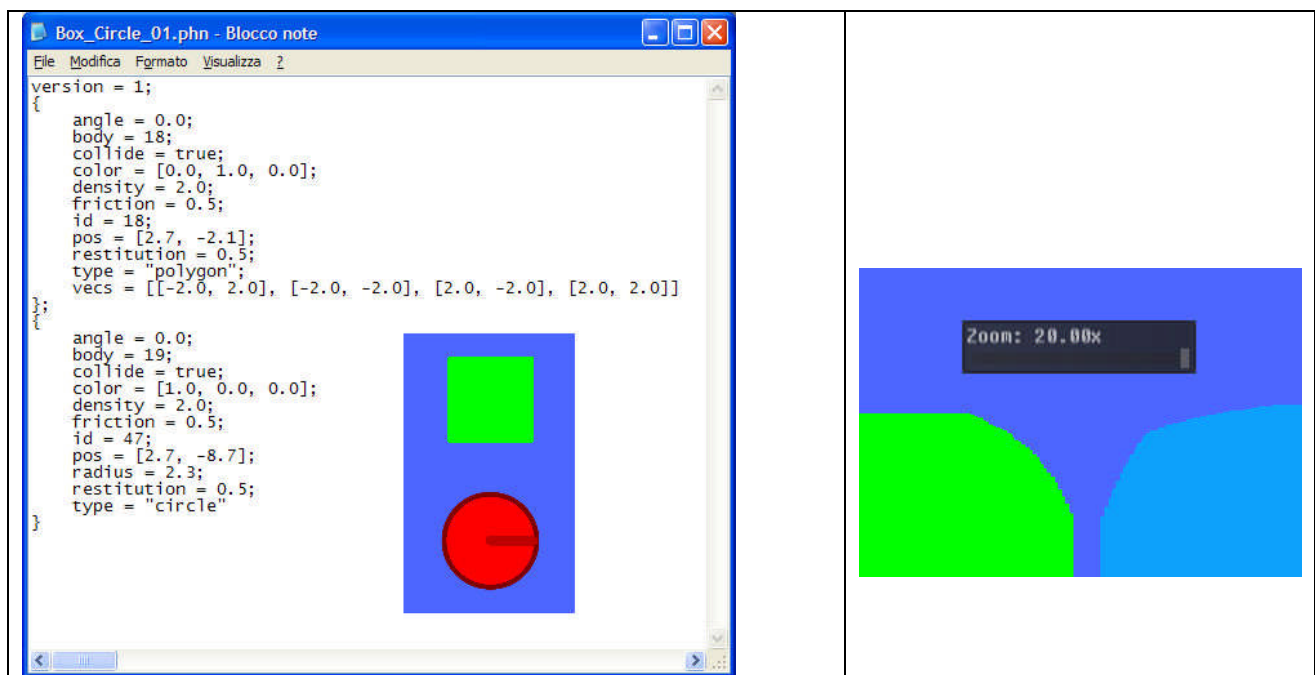
It’s easy to detect the two “blocks” inside the { } parenthesis. According to the specific scene file there are different codes:

Code	Value	Comment
Angle	Decimal value (pos./neg.)	Rotation angle
Body	Entire positive value	???
Collide	True / False	Collision capability
Color	Three decimal positive values	RGB values (Red, Green, Blue)
Density	Decimal positive value	Weigh
Friction	Decimal positive value	
id	Entire positive value	N. object (???)
Pos	Couple decimal values (pos./neg.)	Object's origin coordinates
Restitution	Decimal positive value	???
Type	Polygon / Circe / Spring ...	Object typology
Vecs	Decimal values (pos./neg.)	Vertex's object coordinates
Radius	Decimal positive value	Only for circles

The mean of some codes (see three question marks “???”) is unknown (for me...). Anyway it's possible change file's values.

## Too many parameters

Looking at the previous figure, you can see that the circle has two parameters (origin coordinates and radius) but the rectangle has much more parameters than the origin and four vertexes' coordinates. Anyway you can modify any value. Look at the following figure:



You can see the differences between the vertexes' box at design-time (right) and the box generated by typing values in PHN file (left).

## New horizons

Typing directly values inside the PHN file allows you to realize geometric figures, like triangles. Furthermore, you can perfectly position any figure everywhere you want. Look at the figure below, where two triangle are perfectly positioned inside a box and the circle has the same diameter as the box's height and is strictly tangent to it.

```

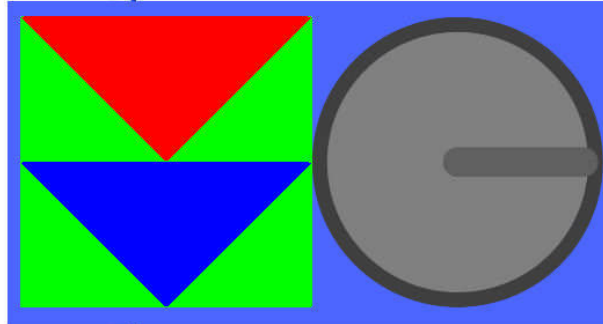
Value_D_2_OK.phn - Blocco note
File Modifica Formato Visualizza 2
version = 1;
{
  angle = 0.0;  Rettangolo verde
  body = 633;
  collide = true;
  color = [0.0, 1.0, 0.0];
  density = 2.0;
  friction = 0.5;
  id = 962;
  pos = [0.0, 0.0];
  restitution = 0.5;
  type = "polygon";
  vecs = [[-2.0, 2.0], [-2.0, -2.0], [2.0, -2.0], [2.0, 2.0]]
};

  angle = 0.0;  Triangolo blu
  body = 634;
  collide = true;
  color = [0.0, 0.0, 1.0];
  density = 2.0;
  friction = 0.5;
  id = 963;
  pos = [0.0, -0.67];
  restitution = 0.5;
  type = "polygon";
  vecs = [[-2.0, 0.0], [0.0, -2.0], [2.0, 0.0]]
};

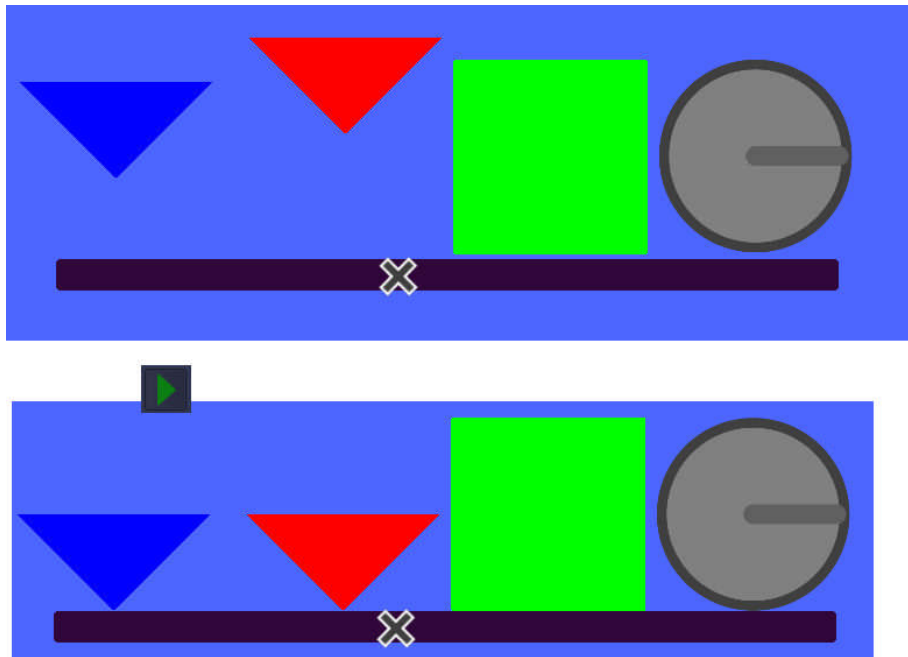
  angle = 0.0;  Triangolo rosso
  body = 635;
  collide = true;
  color = [1.0, 0.0, 0.0];
  density = 2.0;
  friction = 0.5;
  id = 964;
  pos = [0.0, 1.33];
  restitution = 0.5;
  type = "polygon";
  vecs = [[-2.0, 0.0], [0.0, -2.0], [2.0, 0.0]]
};

  angle = 0.0;  Cerchio grigio
  body = 636;
  collide = true;
  color = [0.5, 0.5, 0.5];
  density = 2.0;
  friction = 0.5;
  id = 965;
  pos = [4.0, 0.0];
  radius = 2.0;
  restitution = 0.5;
  type = "circle"
};

```



But more! The design's symmetry pose the objects in perfect balance: try to move objects like in figure below, than press play. The triangles fall down, but they don't rotate and stay equilibrate on inferior vertex!



Il presente Tutorial è stato creato il giorno 9 marzo 2008  
 Per ulteriori aggiornamenti consultare il sito [www.alessandrodesimone.net](http://www.alessandrodesimone.net)